

# 내 손안의 강남상권

강남 외식업 상권분석 웹사이트  
<http://sds10226.cafe24.com/>



**[지원자]**  
강한결

**[팀원]**  
강한결 이영기  
김다솔 김상현  
신동수 방기승  
김정은



# 목 차

---

## 001

### 개요

- 01 | 소 개
- 02 | 기 능

## 002

### 기능 구현

- 01 | 주요 기능

## 003

### 설계

- 01 | DB 스키마
- 02 | Folder 구조
- 03 | Spring 구조

## 004

### 기타

- 01 | 개발 환경
- 02 | 주요 기술
- 03 | 참고
- 04 | 소감





001

개요



## 1. 개요

---

### 01 | 소개

### 02 | 기능

# 프로젝트 명 : 내 손안의 강남상권

‘소상공인마당’의 상권정보시스템

(<http://sg.sbiz.or.kr/main.sg#/main>) 웹 사이트를 모티브로 한

**강남 외식업 상권분석 웹사이트 제작**

## 1. 개요

01 | 소 개

02 | 기 능

## 구현 기능 목록

1. 메인 화면 설계 / 제작
2. 지역 및 업종 검색
3. 회원 가입, 수정, 탈퇴
4. 로그인, 로그아웃
5. 아이디, 비밀번호 찾기
6. 검색 기록 자동 저장
7. 검색리스트 보기
8. 상권평가
9. 업종분석
10. 매출분석
11. 인구분석
12. 지역 분석

\* 담당 업무





002

기능구현



## 2. 기능 구현

### 01 | 주요기능

#### 가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

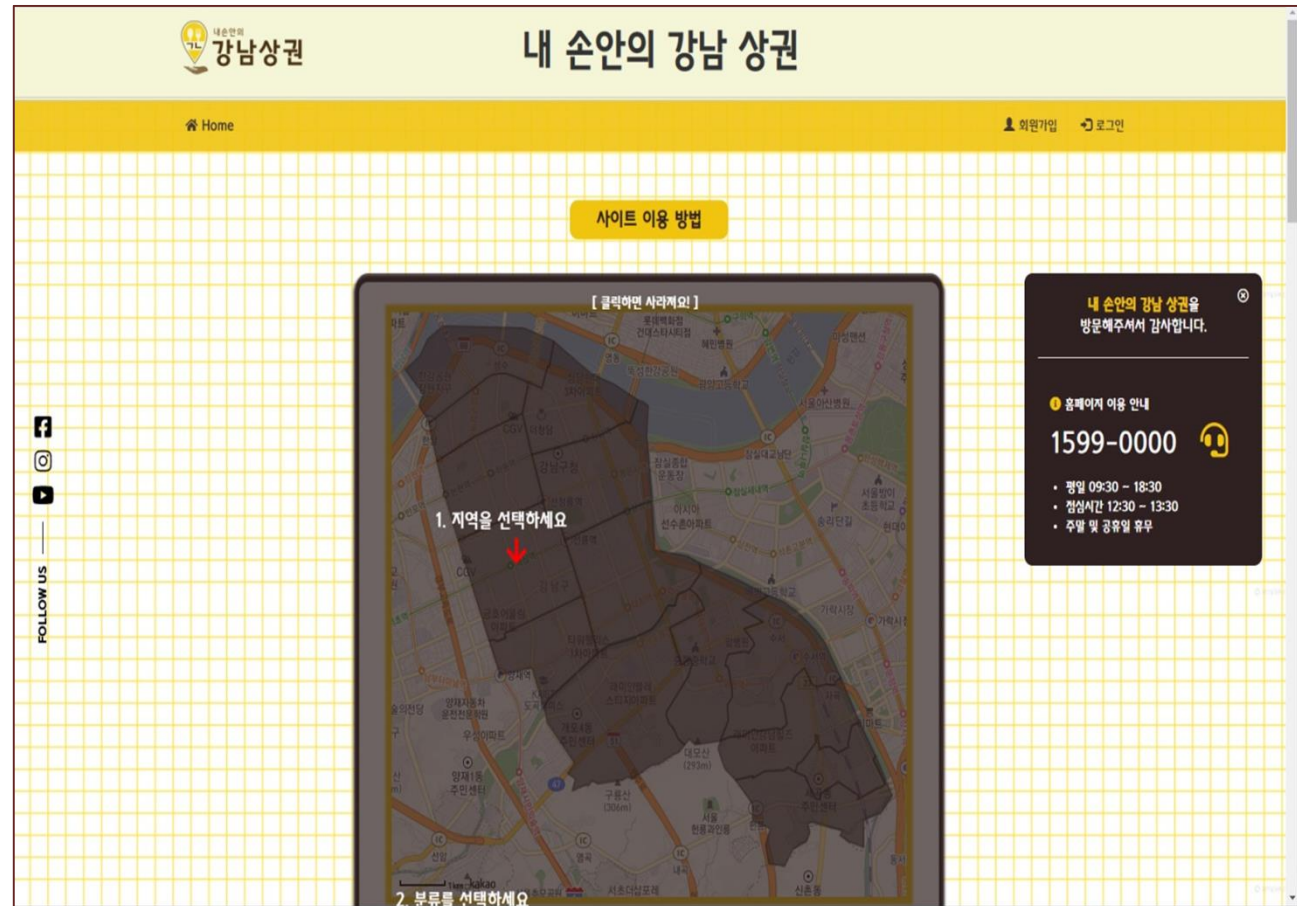
라. 로그인, 로그아웃

마. 아이디, 비밀번호 찾기

바. 검색 기록 자동 저장

사. 검색리스트 보기

## 가. 메인 화면 설계 / 제작



- 간단하게 구현해본 메인 페이지 중 주요 화면 모습입니다.
- 중앙의 지도는 **Kakao api**를 활용하였고, 각 **행정동별 구역**은 **경도와 위도를 배열**에 담아 생성하였습니다.



## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

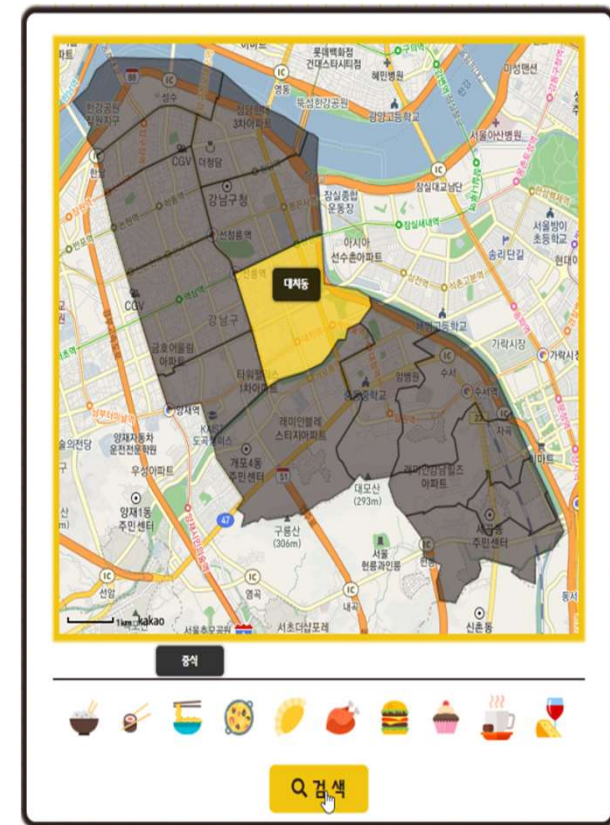
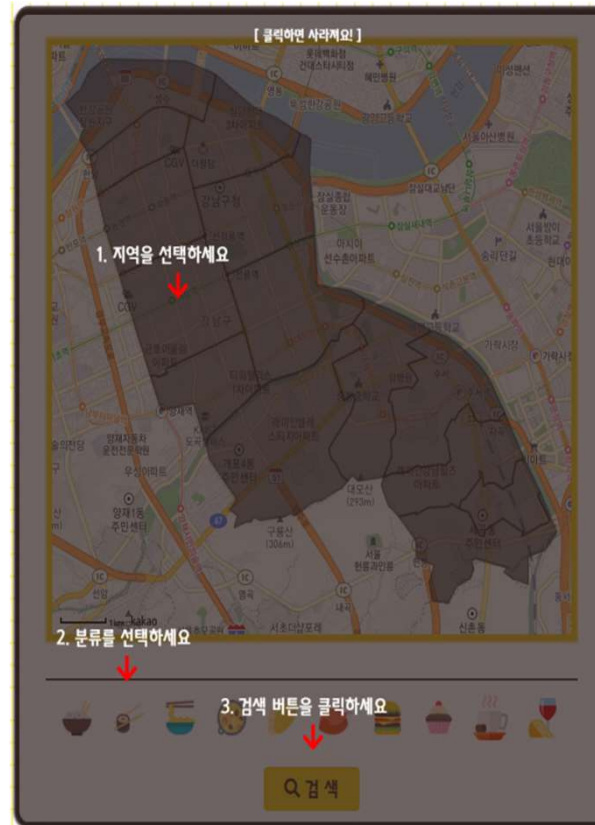
라. 로그인, 로그아웃

마. 아이디, 비밀번호 찾기

바. 검색 기록 자동 저장

사. 검색리스트 보기

## 나. 지역 및 업종 검색



- Stylesheet의 **z-index** 속성을 이용하여 '사용 안내 판'을 검색도구 위에 올려놓았습니다.
- 사용자의 편리함을 위하여 안내 판을 직접 클릭해도 사라지도록 구현하였습니다.
- 검색 버튼을 클릭 시 사용자가 선택한 값을 Parameter로 데이터 분석 페이지에 보냅니다.



## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

라. 로그인, 로그아웃

마. 아이디, 비밀번호 찾기

바. 검색 기록 자동 저장

사. 검색리스트 보기

## 다. 회원가입, 수정, 탈퇴

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE configuration
3 PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4 "http://mybatis.org/dtd/mybatis-3-config.dtd">
5 <configuration>
6   <typeAliases>
7     <!-- 회원 관리 -->
8     <typeAlias alias="memberdto" type="com.cafe24.sosang.dto.MemberDTO"/>
9     <typeAlias alias="mylogdto" type="com.cafe24.sosang.dto.MyLogDTO"/>
10  </typeAliases>
11
12   <mappers>
13     <!-- 회원 관리 -->
14     <mapper resource="mappings/member-mapping.xml"/>
15  </mappers>
16 </configuration>
```

- Mybatis의 기능을 활용하여 SQL 사용에 편리함을 얻기 위해 member-mapping.xml을 mapping하였습니다.
- DTO 클래스의 Full name을 매번 작성하지 않기 위해 typeAlias 설정을 해주었습니다.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper
3 PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
5
6   <!-- 회원가입 : 비밀번호 암호화 -->
7   <insert id="insertMember" parameterType="memberDTO">
8     insert into customer
9     (name, tel, id, password, email)
10    values (#{name}, #{tel}, #{id},
11            <u>#{password}</u>, #{email})
12  </insert>
```



Database
password
7BE7AA1DBAB76405CD2678325102A28E
40581A195E15099AA6C43755F47CC899
A7564DBEFA4B17B4B51F51764F31FF7C
692527347A0F48F373C5F86E22CAC9A2
B77F711C5FE7FB031B1CFA3FC41DD09E
8686F306AA9D5DB42558C672F64F7D63

- 회원 가입 시 비밀번호를 16진수(HEX)로 암호화(AES\_ENCRYPT)하여 Database에 저장함으로 보안을 강화했습니다.

## 2. 기능 구현

### 01 | 주요기능

- 가. 메인 화면 설계 / 제작
- 나. 지역 및 업종 검색
- 다. 회원 가입, 수정, 탈퇴**
- 라. 로그인, 로그아웃
- 마. 아이디, 비밀번호 찾기
- 바. 검색 기록 자동 저장
- 사. 검색리스트 보기

## 다. 회원가입, 수정, 탈퇴



- '마이페이지'에서 회원정보 페이지에 접근하기 위해서는 사용자 비밀번호를 다시 한 번 입력하도록 구현하였습니다.



## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

라. 로그인, 로그아웃

마. 아이디, 비밀번호 찾기

바. 검색 기록 자동 저장

사. 검색리스트 보기

## 다. 회원가입, 수정, 탈퇴

- 1. 사용자 비밀번호를 변경하지 않았을 때 | 2. 사용자 비밀번호를 변경하였을 때  
두 가지 경우의 기능을 한 페이지에서 처리할 수 있도록 구현하였습니다.
- 사용자가 비밀번호를 변경할 경우에도 비밀번호를 암호화하여 DB에 저장하게 하였습니다.

#### changeinfo.js

```
// 비밀번호를 변경 하지 않았을 때와 변경 했을 때를 분리
if(memberChangeInfoForm.confirmPassword.value) {
    memberChangeInfoForm.action = "update_memberWithPassword.do";
    memberChangeInfoForm.submit();
} else if(!memberChangeInfoForm.confirmPassword.value) {
    memberChangeInfoForm.action = "update_member.do";
    memberChangeInfoForm.submit();
}
```

#### MemberController.java

```
// 정보수정 : 비밀번호 제외
@RequestMapping(value = "update_member.do", method = RequestMethod.POST)
public ModelAndView update_member(MemberDTO dto, ModelAndView mav) throws
{
    System.out.println("MemberController-개인정보 수정 : 비밀번호 제외");
    memberService.updateMember(dto);
    mav.setViewName("member/memberUpdateComplete.jsp");
    return mav;
}
```

```
// 정보수정 : 비밀번호 포함
@RequestMapping(value = "update_memberWithPassword.do", method =
public ModelAndView update_memberWithPassword(MemberDTO dto, Mode
    System.out.println("MemberController-개인정보 수정 : 비밀번호 포함");
    memberService.updateMember(dto);
    mav.setViewName("member/memberUpdateComplete.jsp");
    return mav;
}
```

#### member-mapping.xml

```
<!-- 회원정보 업데이트 : 비밀번호 제외 -->
<update id="updateMember" parameterType="memberDTO">
    update customer
    set name =#{name}, tel=#{tel}, email=#{email}
    where id = #{id}
</update>
```

```
<!-- 회원정보 업데이트 : 비밀번호 포함 : 비밀번호 암호화 -->
<update id="updateMemberWithPassword" parameterType="memberDTO">
    update customer
    set name =#{name}, tel=#{tel},
    password=(HEX(AES_ENCRYPT(#{password}, #{id}))), email=#{email}
    where id = #{id}
</update>
```

## 2. 기능 구현

### 01 | 주요기능

- 가. 메인 화면 설계 / 제작
- 나. 지역 및 업종 검색
- 다. 회원 가입, 수정, 탈퇴**
- 라. 로그인, 로그아웃
- 마. 아이디, 비밀번호 찾기
- 바. 검색 기록 자동 저장
- 사. 검색리스트 보기

## 다. 회원가입, 수정, 탈퇴



```
MemberController.java
// 회원탈퇴
@RequestMapping(value = "deleteMember.do")
public ModelAndView deleteMember(MemberDTO dto, ModelAndView mav) {
    logger.info("controller-deleteMember연결");

    dto.setId((String) session.getAttribute("id"));
    memberService.deleteMember(dto);
    memberService.logout(session);

    mav.addObject("message", "다음에 또 만나요!");
    mav.setViewName("member/memberDeleteComplete.jsp");
    return mav;
}
```

```
MemberDAOImpl.java
@Override // 회원탈퇴
public void deleteMember(MemberDTO dto) {
    sqlSession.delete(namespace + ".deleteMember", dto);
    sqlSession.delete(namespace + ".deleteLog", dto);
}
```

```
member-mapping.xml
<!-- 회원탈퇴 -->
<delete id="deleteMember" parameterType="memberDTO">
    DELETE FROM customer
    where id = #{id}
</delete>

<!-- 회원탈퇴 : 로그 삭제 -->
<delete id="deleteLog" parameterType="memberDTO">
    DELETE FROM mylog
    where id = #{id}
</delete>
```

- 회원 탈퇴 시 뒷부분에 나올 '검색 기록'에서도 등록된 데이터를 삭제하도록 하였습니다.



## 2. 기능 구현

### 01 | 주요기능

- 가. 메인 화면 설계 / 제작
- 나. 지역 및 업종 검색
- 다. 회원 가입, 수정, 탈퇴
- 라. 로그인, 로그아웃**
- 마. 아이디, 비밀번호 찾기
- 바. 검색 기록 자동 저장
- 사. 검색리스트 보기

## 라. 로그인, 로그아웃

```
MemberController.java
// 로그인
@RequestMapping(value = "login_check.do", method = RequestMethod.POST)
public String login_check(MemberDTO dto, Model model, HttpSession session) {
    System.out.println("MemberController-로그인");
    MemberDTO member_info = memberService.login(dto, session);
    if (member_info != null) {
        return "redirect:../home.jsp";
    } else { // 로그인이 실패하면 login폼으로 돌아간다.
        model.addAttribute("message", "로그인에 실패했습니다.");
        return "member/loginForm.jsp";
    }
}
```

```
MemberServiceImpl.java
@Override // 로그인
public MemberDTO login(MemberDTO dto, HttpSession session) {
    MemberDTO member_info = memberDao.login(dto);
    if (member_info != null) { // 아이디와 비밀번호가 맞은 경우
        // 세션변수 등록
        session.setAttribute("id", dto.getId());
    }
    return member_info;
}
```

```
member-mapping.xml
<!-- 로그인 : 비밀번호 복호화 -->
<select id="login_check" parameterType="memberDTO" resultType="memberDTO">
    select name, tel, id, password, email
    from customer
    where id = #{id} and AES_DECRYPT(UNHEX(password), #{id}) = #{password}
</select>
```

- 로그인 시 DB에 입력된 비밀번호를 복호화 하여 사용자가 입력한 비밀번호와 비교합니다.
- 사용자가 입력한 아이디, 비밀번호에 맞는 데이터가 DB에 있다면 로그인하고 session을 발급하도록 구현하였습니다.

## 2. 기능 구현

### 01 | 주요기능

- 가. 메인 화면 설계 / 제작
- 나. 지역 및 업종 검색
- 다. 회원 가입, 수정, 탈퇴
- 라. 로그인, 로그아웃
- 마. 아이디, 비밀번호 찾기**
- 바. 검색 기록 자동 저장
- 사. 검색리스트 보기

## 마. 아이디, 비밀번호 찾기

```
MemberController.java
// 아이디, 비밀번호 찾기
@RequestMapping(value = "find_Id.do")
public ModelAndView find_id(MemberDTO dto, HttpServletRequest req) {
    System.out.println("MemberController-아이디 찾기");
    dto.setName((String) req.getParameter("name"));
    dto.setEmail((String) req.getParameter("email"));

    // ModelAndView 만드기
    mav.addObject("findId", memberService.findUserId(dto));
    mav.setViewName("member/find_Id.jsp");
    return mav;
}

<c:set var="findId" value="${findId }" />
<c:choose>
    <c:when test="${empty findId }">
        <div class="resIdBox">
            <span class="resId">해당하는 데이터가 없습니다.</span>
        </div>
    </c:when>
    <c:otherwise>
        <div class="resIdBox">
            <span class="resId">아이디 찾기 결과 : ${findId }</span>
        </div>
    </c:otherwise>
</c:choose>
```

- 아이디 찾기 결과를 간단한 JSTL 문법을 활용해 팝업창에서 안내하도록 구현하였습니다.



## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

라. 로그인, 로그아웃

**마. 아이디, 비밀번호 찾기**

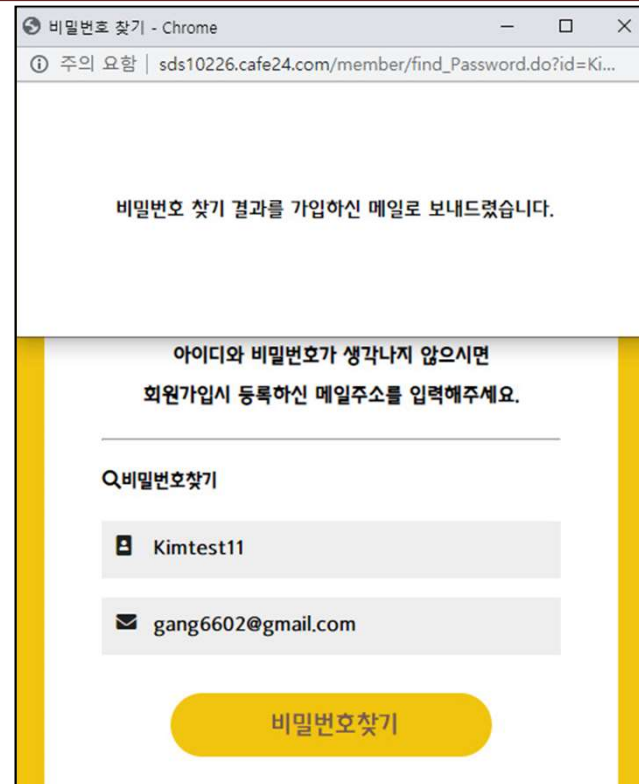
1) 비밀번호 찾기

2) Email로 보내기

바. 검색 기록 자동 저장

사. 검색리스트 보기

## 마. 아이디, 비밀번호 찾기



```
<!-- 비밀번호 찾기 : 비밀번호 복호화 -->
<select id="find_Password" parameterType="java.util.Map" resultType="String">
    select AES_DECRYPT(UNHEX(password), #{id})
    from customer
    where id=#{id} and email=#{email}
</select>
```

- 비밀번호 찾기 시 암호화된 비밀번호를 복호화하여 사용자가 회원가입할 때 입력한 이메일로 비밀번호를 보냅니다.

## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

라. 로그인, 로그아웃

마. 아이디, 비밀번호 찾기

1) 비밀번호 찾기

2) Email로 보내기

바. 검색 기록 자동 저장

사. 검색리스트 보기

## 마. 아이디, 비밀번호 찾기

Sosang1.7/pom.xml

```
<!-- java : email 기능(아이디/비밀번호 찾기) -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context-support</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
<dependency>
  <groupId>javax.mail</groupId>
  <artifactId>mail</artifactId>
  <version>1.4.7</version>
</dependency>
```

com.cafe24.sosang.email

- Email.java
- EmailSender.java

```
import javax.mail.MessagingException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.MailException;
import org.springframework.mail.javamail.JavaMailSender;

public class EmailSender {

    @Autowired
    protected JavaMailSender mailSender;

    public void SendEmail(Email email) throws Exception {

        MimeMessage msg = mailSender.createMimeMessage();

        try {
            msg.setSubject(email.getSubject());
            msg.setText(email.getText());
            msg.setRecipients(MimeMessage.RecipientType.TO,
                InternetAddress.parse(email.getReceiver()));
        } catch (MessagingException e) {
            e.printStackTrace();
        }

        try {
            mailSender.send(msg);
        } catch (MailException e) {
            e.printStackTrace();
        }

    }
}
```

- 이메일 기능을 사용하기 위해 **pom.xml**에 각각의 기능을 **dependency**로 묶어서 라이브러리가 **Maven**에 추가될 수 있도록 하였습니다.
- DTO** 기능의 **Email Class**와 **Process**를 담당하는 **EmailSender Class**를 생성하였습니다.

## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

라. 로그인, 로그아웃

**마. 아이디, 비밀번호 찾기**

1) 비밀번호 찾기

2) Email로 보내기

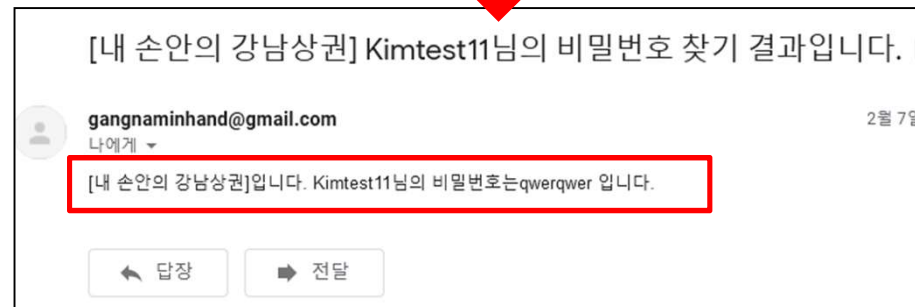
바. 검색 기록 자동 저장

사. 검색리스트 보기

## 마. 아이디, 비밀번호 찾기

```
MemberController.java
// 비밀번호 찾기시 결과를 사용자 이메일로 보냄
@RequestMapping(value = "find_Password.do")
public ModelAndView find_Password(@RequestParam Map<String, Object> paramMap, Model model) {
    System.out.println("MemberController-비밀번호 찾기");
    String id = (String) paramMap.get("id");
    String customer_mail = (String) paramMap.get("email");

    String pw = memberService.findUserPassword(paramMap);
    if(pw != null) {
        email.setText("[내 손안의 강남상권]입니다. " + id + "님의 비밀번호는" + pw + " 입니다.");
        email.setReceiver(customer_mail);
        email.setSubject("[내 손안의 강남상권] " + id + "님의 비밀번호 찾기 결과입니다.");
        emailSender.SendEmail(email);
        mav.addObject("findPassword", memberService.findUserPassword(paramMap));
        mav.setViewName("member/find_Password.jsp");
    }
    mav.setViewName("member/find_Password.jsp");
    return mav;
}
```



- 비밀번호 찾기 후 실제로 받게 되는 메일입니다.



## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

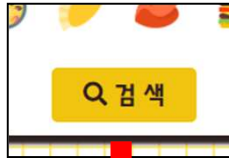
라. 로그인, 로그아웃

마. 아이디, 비밀번호 찾기

**바. 검색 기록 자동 저장**

사. 검색리스트 보기

## 바. 검색 기록 자동 저장



```
home.jsp
// 선택사항 유효성 검사
function validationChk(mainSubmitForm) {
    if(!mainSubmitForm.area.value) {
        alert('동을 선택해주세요.');
```

→

```
        return false;
    }
    if(!mainSubmitForm.food.value) {
        alert('외식업 종류를 선택해주세요.');
```

→

```
        return false;
    }

    // 유효성 검사 이후 jsp로 parameter submit 하기
    mainSubmitForm.action = "analysis/showData.do";
    mainSubmitForm.submit();
}
```

```
AnalysisControl.java
// 분석페이지 이동
@RequestMapping(value = "/showData.do")
public String showData(AreaDTO adto, MylogDTO mldto, HttpSession session) throws Exception {
    System.out.println("analysisControl => 분석 페이지로 이동");
    session.setAttribute("area", adto.getArea());
    session.setAttribute("food", adto.getFood());

    String id = (String) session.getAttribute("id");
    if(id != null && id != "") {
        // 검색 시간에 저장할 값 구하기
        Timestamp reg_date = new Timestamp(System.currentTimeMillis());

        // mldto에 있는거 다 가져와서 dao에서 mylog table에 저장.
        mldto.setId((String) session.getAttribute("id"));
        mldto.setArea(adto.getArea());
        mldto.setFood(adto.getFood());
        mldto.setSearch_date(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(reg_date));
        memberService.reg_mylog(mldto);
    }
    return "anaysisMenu.jsp";
}
```

- 검색 기록 자동 저장 기능은 회원가입 시 얻게 되는 혜택입니다.
- Session에 id 저장 유무를 확인 후 이를 조건으로 home.jsp에서 검색 버튼을 누르면 검색 기록이 자동 저장되게 구현하였습니다.

member-mapping...

```
<!-- 검색 정보 저장 -->
<insert id="reg_mylog" parameterType="mylogdto">
    insert into mylog
        (id, area, food, search_date)
    values(#{id}, #{area}, #{food}, #{search_date})
</insert>
```

Field	Type	Null	Key	Default	Extra
idx	int(11)	NO	PRI	NULL	auto_increment
id	varchar(16)	NO		NULL	
area	varchar(16)	NO		NULL	
food	varchar(20)	NO		NULL	
search_date	varchar(20)	NO		NULL	

- SimpleDateFormat의 format() 함수를 활용할 경우 날짜가 String 타입으로 변하기 때문에 search\_date의 Type을 varchar로 선언하였습니다.

## 2. 기능 구현

### 01 | 주요기능

가. 메인 화면 설계 / 제작

나. 지역 및 업종 검색

다. 회원 가입, 수정, 탈퇴

라. 로그인, 로그아웃

마. 아이디, 비밀번호 찾기

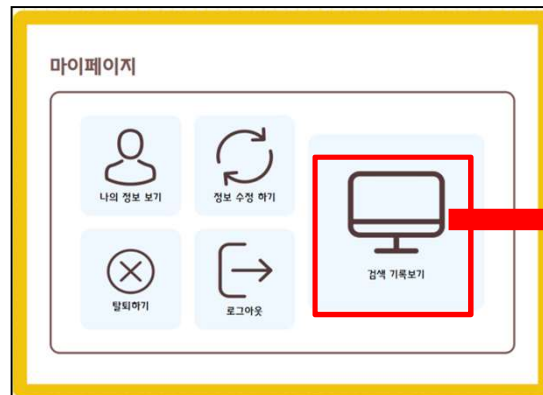
바. 검색 기록 자동 저장

사. 검색리스트 보기

1) 화면 구현

2) 출력 및 삭제

## 사. 검색리스트 보기



Kimtest11님의 검색이력

\* 최근 10개 항목만 표시됩니다.

번호	검색 날짜	지역/동 이름	외식업종	기록보기	삭제
1	2020-02-10 03:30:34	대치동	치킨전문점	보기	삭제
2	2020-02-07 15:56:23	논현동	분식전문점	보기	삭제
3	2020-02-07 15:56:18	도곡동	호프·간이주점	보기	삭제
4	2020-02-07 15:56:12	알원동	치킨전문점	보기	삭제
5	2020-02-07 15:55:45	신사동	한식음식점	보기	삭제
6	2020-02-07 15:55:35	압구정동	분식전문점	보기	삭제
7	2020-02-07 15:55:29	수서동	양식음식점	보기	삭제
8	2020-02-07 15:55:23	압구정동	중식음식점	보기	삭제
9	2020-02-07 15:55:18	도곡동	패스트푸드점	보기	삭제
10	2020-02-07 15:55:12	대치동	분식전문점	보기	삭제

Home My Page

- 마이페이지에서 검색기록보기 버튼 클릭 시 출력되는 화면입니다.
- 최근 10개 항목만 출력하도록 구현하였습니다.

## 2. 기능 구현

### 01 | 주요기능

- 가. 메인 화면 설계 / 제작
- 나. 지역 및 업종 검색
- 다. 회원 가입, 수정, 탈퇴
- 라. 로그인, 로그아웃
- 마. 아이디, 비밀번호 찾기
- 바. 검색 기록 자동 저장

#### 사. 검색리스트 보기

##### 1) 화면 구현

##### 2) 출력 및 삭제

## 사. 검색리스트 보기

### mylog.jsp

```
<c:forEach items="${list}" var="list" varStatus="status" begin="0" end="9">
  <tr>
    <td>${status.count}</td>
    <td>${list.search_date}</td>
    <td>${list.area}</td>
    <td>${list.food}</td>
    <td>
      <a href=" ../analysis/myLogData.do?area=
        ${list.area}&food=${list.food}&id=<%=id %>"> 보기</a>
    </td>
    <td>
      <input type="button" id="deleteLog" value="삭제"
        onclick="deleteLog(${list.idx})">
    </td>
  </tr>
</c:forEach>
```

```
// 내가 검색한 내역에서 보기 버튼 눌렀을 때
@RequestMapping(value = "/myLogData.do")
public String showData(HttpSession session, AreaDTO adto) throws
    System.out.println("analysisControl => Mylog에서 보기 버튼 클릭");
    session.setAttribute("area", adto.getArea());
    session.setAttribute("food", adto.getFood());
    return "anaysisMenu.jsp";
}
```

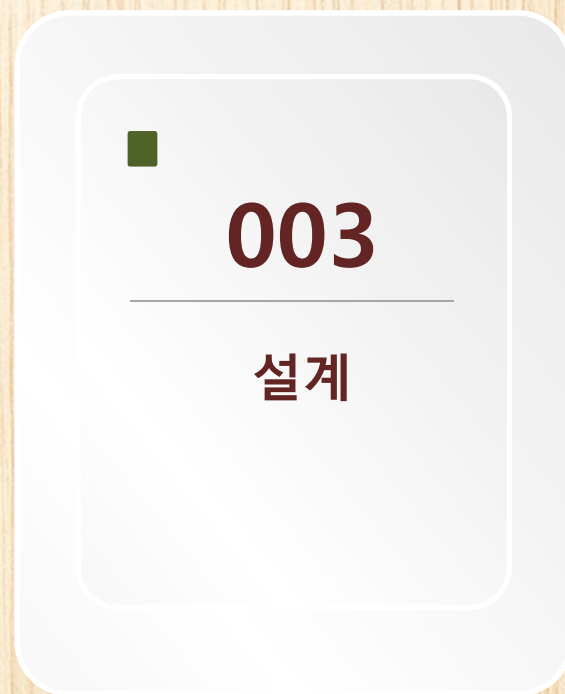
- '보기' 버튼 클릭 시 저장 된 검색 기록을 Parameter로 보내서 '재 검색' 하는 방식으로 구현하였습니다.

### mylog.jsp

```
<script>
  function deleteLog(idx) {
    var delete_confirm = confirm("삭제하시겠습니까?");
    if(delete_confirm == true) {
      window.location.href = 'deleteLog.do?idx=' + idx;
    } else {
      window.location.href = 'showList.do';
    }
  }
</script>
```

- '삭제' 버튼 클릭 시 JS에서 comfirm으로 삭제 여부를 다시 한 번 물어보고, 반환되는 true/false값으로 조건을 생성하였습니다.





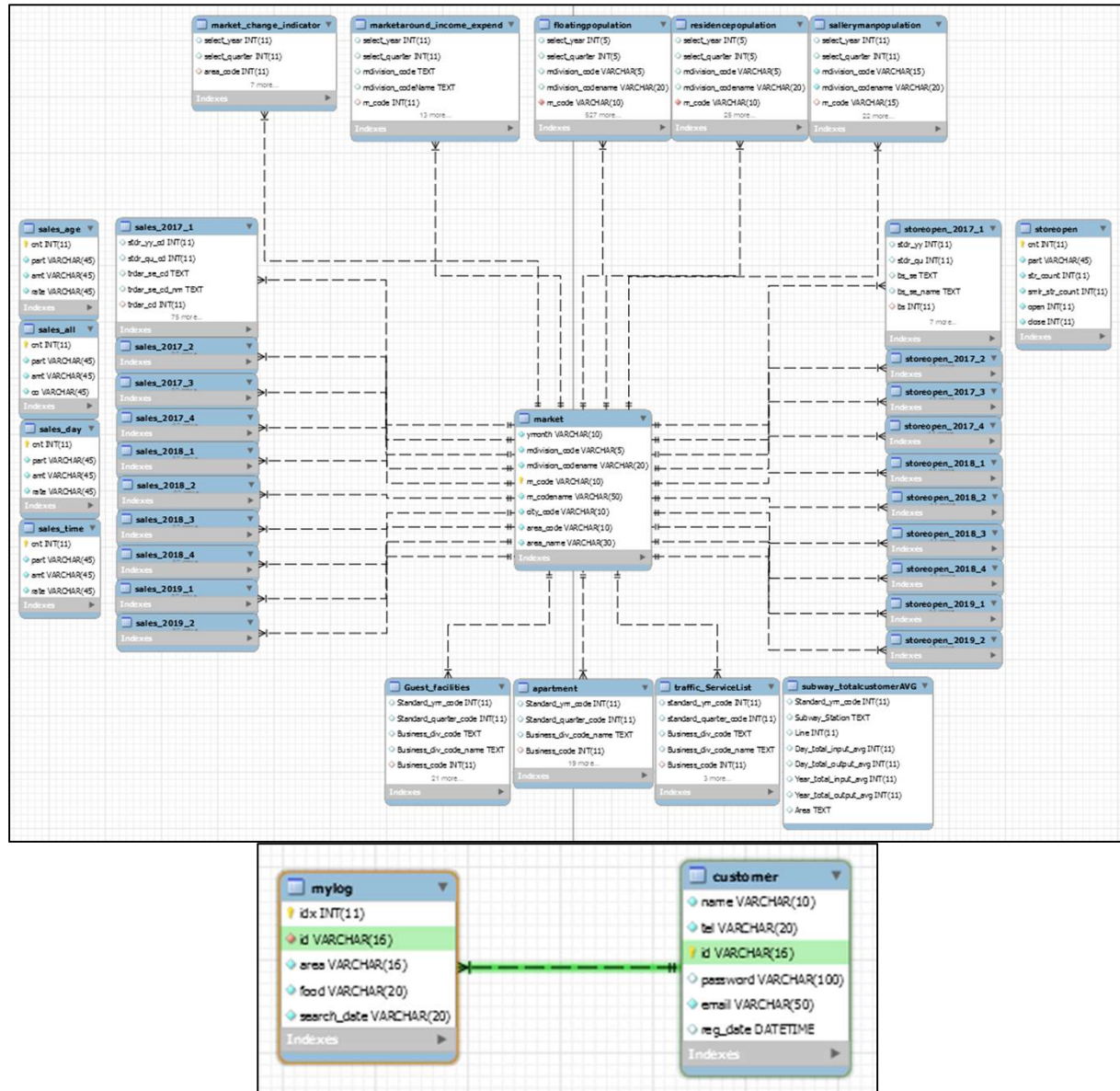
### 3. 설계

#### 01 | DB 스키마

#### 02 | Folder 구조

#### 03 | Spring 구조

## 데이터베이스 개체-관계 모델(ERD)





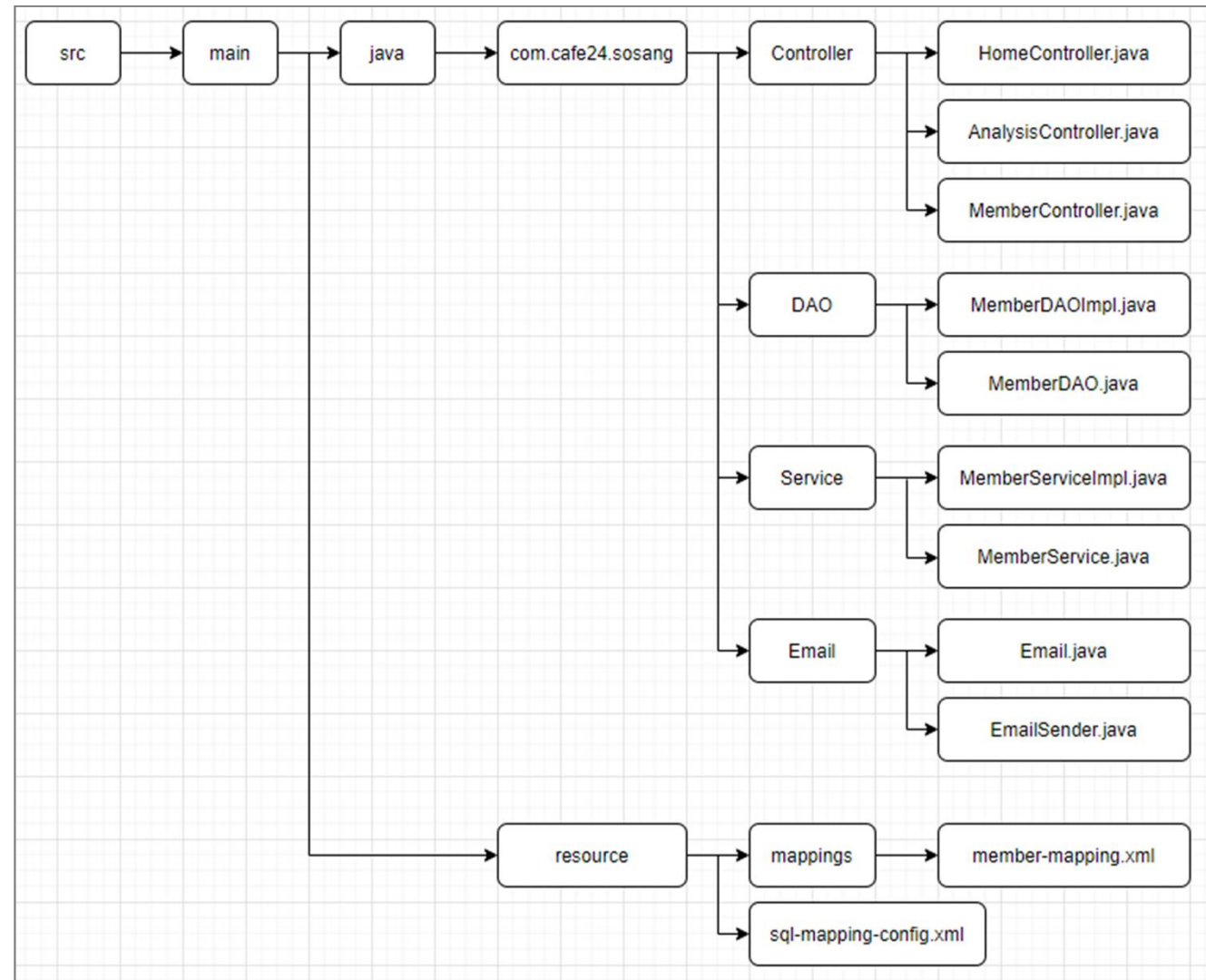
### 3. 설계

01 | DB 스키마

**02 | Folder 구조**

03 | Spring 구조

## Java, xml 파일 경로





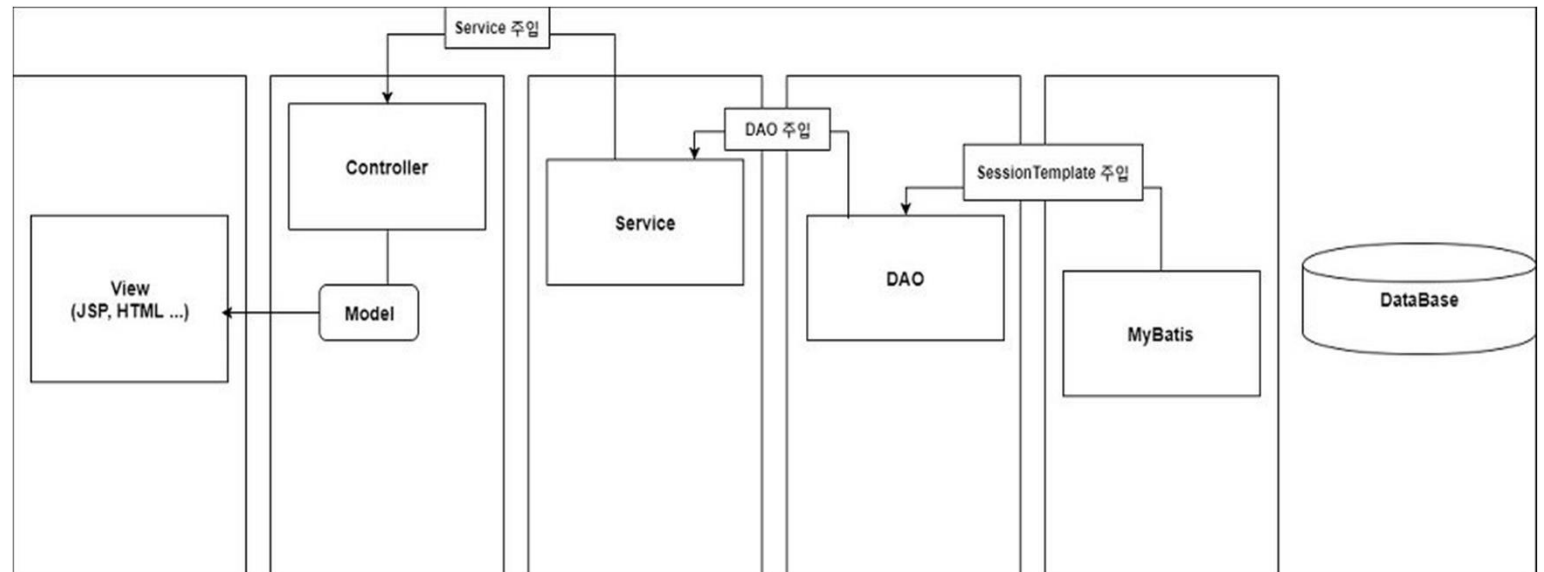
### 3. 설계

01 | DB 스키마

02 | Folder 구조

**03 | Spring 구조**

## Spring MVC 흐름도





004

기타



## 4. 기타

---

### 01 | 개발 환경

02 | 주요 기술

03 | 참고

04 | 소감

## 개발 환경

### 1. 개발 언어

- Java 8
- JSP
- JavaScript

### 2. 개발 도구

- Spring Tool Suite 4
- Visual Studio Code
- MySQL

### 3. 개발 환경

- Window 10
- Apache Tomcat 8.5

### 4. 개발 H/W

- PC



## 4. 기타

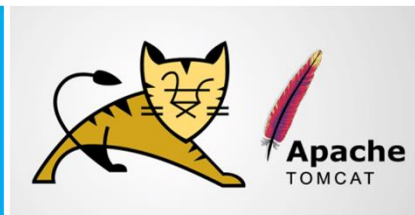
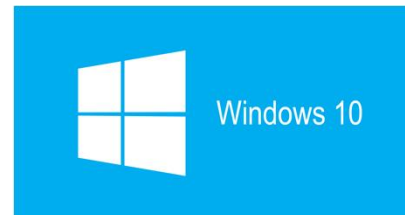
01 | 개발 환경

**02 | 주요 기술**

03 | 참고

04 | 소감

### 주요 기술



## 4. 기타

---

01 | 개발 환경

02 | 주요 기술

**03 | 참고**

04 | 소감

### 참고 : 기준 데이터

- 서울 열린데이터 광장  
(<https://data.seoul.go.kr/dataList/datasetList.do>)  
서울시 우리마을가게 상권분석 서비스 사용

## 4. 기타

01 | 개발 환경

02 | 주요 기술

03 | 참고

**04 | 소감**

### 소감

#### • 어려웠던 점

첫 팀 프로젝트를 MVC model-1 으로 진행했었던 만큼 프로젝트 진행 초반에는 Spring 구조에 적응하기가 다소 힘들었습니다.

하지만 선생님께 Spring 공부에 도움이 될만한 서적을 추천 받아 '스프링 퀵 스타트'를 구매하였고, 틈틈이 이 책을 읽거나 선생님의 소스코드를 분석하여 어느 정도 극복할 수 있었습니다.

#### • 느낀 점

MVC model-1에서의 복잡한 SQL문 사용법이 Mybatis로 너무나도 쉽게 해결되는 것이 매우 편리했고, 이 전 프로젝트에서 구현하고자 했었던 Email 기능이나 사용자 비밀번호 암호화와 같은 기능들을 새롭게 구현해볼 수 있어서 즐거웠습니다.



# 감사합니다

---

gang6607@gmail.com

지원자 : 강 한결 드림